COMP 3512 Assignment #3

Due Midnight-ish Dec 8 - 11
Version 1.0, Oct 31 2016

Overview

You can work in groups of twos/three for this assignment. It is also possible to work individually, but I do strongly discourage it; please talk to me about this if you are planning on working by yourself. This assignment will allow you to apply the concepts and technologies covered in class to a "real" project situation. In this project, you will expand your second assignment using JavaScript and jQuery (anywhere this assignment says do some task using JavaScript, you can also use jQuery).

If working in a group, each member needs to take responsibility for and complete an appropriate amount of the project work. **Be sure to consult the instructor at least one week prior to the due date if your group is experiencing serious problems in this regard.**

Submitting

Put your assignment in a folder named assign3_yourlogin (e.g., assign3_fsmi9876). Only one person in the group needs to submit the assignment. You must also make this site viewable online. I would recommend using Cloud9 or CodeAnywhere (though some you might decide to try something more adventurous such as Heroku or some other web hosting environment). You must then share the URL with me in some manner. If using Cloud9, for instance, you can share your assignment Workspace with me. If using some other hosting option, send me an email with a link that allows me to view your assignment.

Put all resources used by your assignment into this folder. Copy this folder to the submit drive in B215 and to the normal university-wide submit drive. You will lose marks if you do not follow these submission instructions.

Grading

The grade for this assignment will be broken down as follows:

Usability and visual design	20%
Program design **and** documentation	15%
Features	65%

USE CASE NAME:	**Assignment 2 Functionality**
DESCRIPTION:	Expected base functionality
1.	Your system must have ALL the functionality from assignment 2.

USE CASE NAME:	**Browse Painting Preview**
DESCRIPTION:	Way to preview painting from thumbnail image
1.	Initiated when user moves mouse cursor over a smaller version of a painting (i.e., in Browse Painting and in painting lists in single genre, single artist, and single gallery).
2.	The system will use JavaScript to display the image in a dynamically-generated pop-over `<div>`. The image that should be displayed is the appropriate one from the `average` folder.
3.	When the user moves the mouse out of the painting image, the generated pop-over should disappear.

USE CASE NAME:	**View Cart**
DESCRIPTION:	Use JavaScript to make cart more interactive
1.	In Assignment 2, all the calculation logic of the cart was performed on the server-side using PHP. In this assignment, you will perform the calculations using JavaScript. Whenever the user changes an element in the form, the totals should change directly without any server-side intervention.
2	Remove the update button (unnecessary since JavaScript is performing the calculations now). The Continue Shopping buttons will continue to preserve the user’s choices in session state.

USE CASE NAME:	**Painting Web Service**
DESCRIPTION:	Create a PHP-driven web service to provide a list of paintings in JSON format that matches the specified criteria.
1.	This will be the JSON web service equivalent of the functionality in the Browse Painting page. The service must be called service-painting.php.
2.	If no parameters are passed, it should return all the paintings (limit 20). The possible parameters must be: ● `artist` – return top 20 paintings for the specified artist ID. ● `shape` – return top 20 paintings for the specified shape ID. ● `gallery` – return top 20 paintings for the specified gallery ID. ● `name` – return top 20 paintings whose title **begins** with the specified search string.

USE CASE NAME:	**Browse Painting**
DESCRIPTION:	Asynchronously consume the Painting Web Service instead of using back-end PHP
1.	From assignment one and two, you have a browse painting page that filters the list of paintings when the use clicks the Filter button. Now, you will eliminate the Filter button, and instead display the matching paintings whenever the user selects a new artist, shape, or gallery from the filter drop-down lists using JavaScript and your Painting Web Service.
2	When the user selects a new filter, run the Semantic UI slide left Transition to remove the current painting list, and the use the slide right transition when ready to display new list of paintings.
3	When the user selects a new filter, display a Semantic UI Loader after the slide left transition. Once the data is received from the web server, hide the loader and then perform the slide right transition with the new painting list.

USE CASE NAME:	**Simple Search**
DESCRIPTION:	System allows user to search within the paintings from a single search box.
1.	This use case is initiated when a user enters two or more characters of text into the search box. The system will display paintings whose title **begins** with the entered text using the Semantic UI Search module. Note: in assignment 2, the search looked for the search string anywhere within the title or description. Here it is just the beginning of the title.
2.	The Semantic UI Search module will use your Painting Web Service to display a dynamic drop-down list that displays painting title as the title, and the artist name as the description.
3.	When the user selects an item from the search list, redirect to single-painting.php with the appropriate query string.

USE CASE NAME:	**Search Results**
DESCRIPTION:	No longer needed
1.	Due to the change in simple search, there will no longer be the search result case.